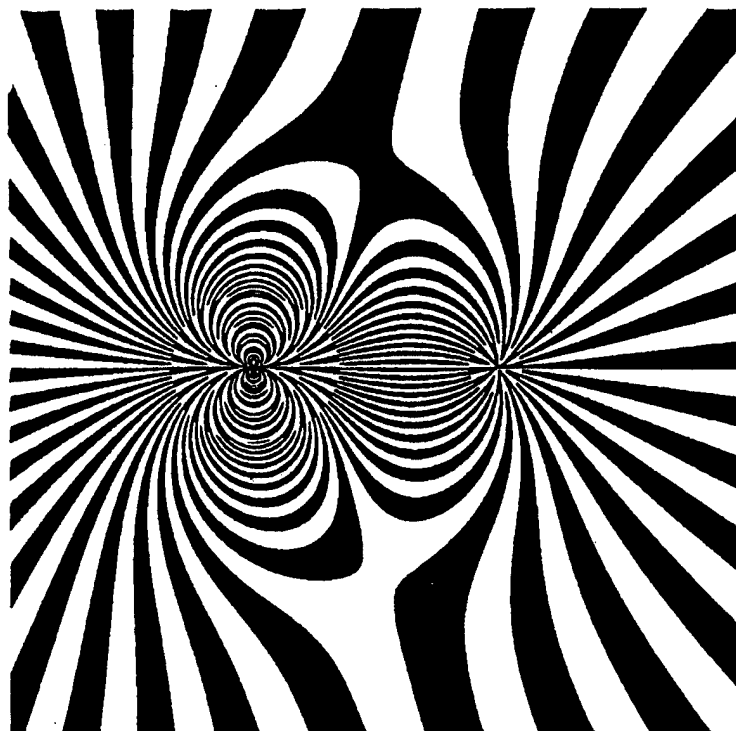# Research on the Causes of Dynamical Instability in Combat Models

by
Julian I. Palmore

Combat models are nonlinear deterministic models of decisionmaking processes that deal with attrition of opposing forces. They contain dynamical instabilities that destroy the robustness of a simulation and interfere with a simulation's consistency, fidelity, reliability, transportability, and validity. Nonlinear behavior in a model's design is a primary source of instability. Striking effects of instabilities on the performance of a combat model are found in computer arithmetic that cause global divergences of simulation runs, branchings on thresholds set by decision tables that can be triggered inadvertently by small errors in the computation of state variables, and structurally unstable decision logic.

Difficulties in reducing and eliminating structural instability and structural variance arise from the use of modeling paradigms. Several paradigms are distinguished: discrete event simulations, embedded dynamical systems, and models of computer arithmetic. A dynamical systems viewpoint illuminates properties of a model and its computer simulation that are essential for verification and validation. By viewing computational processes as dynamical systems that are embedded in discrete event simulations, fundamental difficulties for models of the real world are brought to light.

This report discusses examples of unstable behavior, demonstrates ways to reduce or eliminate sources of instability, and suggests strategies for designing valid combat models that are consistent, robust, and stable.

Phase portrait of logistics flow on plane.

19961008 084

DTIC QUALITY INSPECTED 3

USER EVALUATION OF REPORT

REFERENCE: USACERL Technical Report 96/95, *Research on the Causes of Dynamical Instability in Combat Models*

Please take a few minutes to answer the questions below, tear out this sheet, and return it to USACERL. As user of this report, your customer comments will provide USACERL with information essential for improving future reports.

1. Does this report satisfy a need? (Comment on purpose, related project, or other area of interest for which report will be used.)

_____

_____

_____

2. How, specifically, is the report being used? (Information source, design data or procedure, management procedure, source of ideas, etc.)

_____

_____

3. Has the information in this report led to any quantitative savings as far as manhours/contract dollars saved, operating costs avoided, efficiencies achieved, etc.? If so, please elaborate.

_____

_____

4. What is your evaluation of this report in the following areas?

a. Presentation: _____

b. Completeness: _____

c. Easy to Understand: _____

d. Easy to Implement: _____

e. Adequate Reference Material: _____

f. Relates to Area of Interest: _____

g. Did the report meet your expectations? _____

h. Does the report raise unanswered questions? _____

i. General Comments. (Indicate what you think should be changed to make this report and future reports of this type more responsive to your needs, more usable, improve readability, etc.)

_____

_____

_____

_____

_____ _____

_____

5. If you would like to be contacted by the personnel who prepared this report to raise specific questions or discuss the topic, please fill in the following information.

Name: _____ _____

Telephone Number: _____

Organization Address: _____

_____

_____

6. Please mail the completed form to:

Department of the Army
CONSTRUCTION ENGINEERING RESEARCH LABORATORIES
ATTN: CECER-TR-I
P.O. Box 9005
Champaign, IL 61826-9005

REPORT DOCUMENTATION PAGE

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	August 1996	Final

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Research on the Causes of Dynamical Instability in Combat Models	ILIR A91D-FA-IG2

6. AUTHOR(S)

Julian I. Palmore

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
U.S. Army Construction Engineering Research Laboratories (USACERL) P.O. Box 9005 Champaign, IL 61826-9005	TR 96/95

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING / MONITORING AGENCY REPORT NUMBER
U.S. Army Construction Engineering Research Laboratories (USACERL) P.O. Box 9005 Champaign, IL 61826-9005	

11. SUPPLEMENTARY NOTES

Copies are available from the National Technical Information Service, 5285 Port Royal Road, Springfield, VA 22161.

12a. DISTRIBUTION / AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for public release; distribution is unlimited.	

13. ABSTRACT (Maximum 200 words)

Combat models are nonlinear deterministic models of decisionmaking processes that deal with attrition of opposing forces. They contain dynamical instabilities that destroy the robustness of a simulation and interfere with a simulation's consistency, fidelity, reliability, transportability, and validity. Nonlinear behavior in a model's design is a primary source of instability. Striking effects of instabilities on the performance of a combat model are found in computer arithmetic that cause global divergences of simulation runs, branchings on thresholds set by decision tables that can be triggered inadvertently by small errors in the computation of state variables, and structurally unstable decision logic.

Difficulties in reducing and eliminating structural instability and structural variance arise from the use of modeling paradigms. Several paradigms are distinguished: discrete event simulations, embedded dynamical systems, and models of computer arithmetic. A dynamical systems viewpoint illuminates properties of a model and its computer simulation that are essential for verification and validation. By viewing computational processes as dynamical systems that are embedded in discrete event simulations, fundamental difficulties for models of the real world are brought to light.

This report discusses examples of unstable behavior, demonstrates ways to reduce or eliminate sources of instability, and suggests strategies for designing valid combat models that are consistent, robust, and stable.

14. SUBJECT TERMS		15. NUMBER OF PAGES
modeling computer programs combat	simulation mathematical models	54
		16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	SAR

NSN 7540-01-280-5500

Foreword

Contents

Distribution

1 Introduction

Background

Combat models—nonlinear deterministic models of decisionmaking processes dealing with attrition of opposing forces—contain dynamical instabilities. Instability destroys the robustness of a simulation and interferes with its performance. Measures of robustness are consistency, portability, reliability, reproducibility of results, and validity. Nonlinear behavior in a model's design is a primary source of instability. Nonlinear behavior in computer arithmetic—the basis of calculations in computer simulations—causes global divergences of simulation results in regions where a simulation depends sensitively upon initial conditions. Structural variance, the name given to unexpected divergences of simulation results and peculiar simulation behavior, is directly related to the interaction of several modeling paradigms: discrete event simulation, continuous system simulation, and computer arithmetic. Structural variance manifests itself as unusual sensitivity to input data or logic, and changes resulting from the use of different computers.

A dynamical systems viewpoint illuminates properties of a model and its computer simulation that are essential for verification and validation. By viewing computational processes as dynamical systems that are embedded in discrete event simulations, fundamental difficulties in modeling the real world are brought to light.

Modeling and simulation technologies have many potential applications to the facilities technology and land management missions of the U.S. Army Construction Engineering Research Laboratories (USACERL). As part of its basic research in modeling and simulation enabling technologies, the USACERL Military Engineering Team investigated the problem of dynamical instability in combat models. However, the findings of these investigations of dynamical systems should apply to any other complex models of the real world.

Objectives

The overall objective of the research reported here was to study the causes of dynamical instability in combat models and suggest strategies for reducing or eliminating the sources of structural variance.

The objective of this report is to consolidate the published findings of several related basic research work units into an integrated summary not previously available to the modeling and simulation technical community.

Approach

In mathematics, dynamics addresses time and its representations. This is particularly appropriate in computer simulations of combat, which mix continuous system simulation carrying a globally defined time, with discrete event simulation carrying ordering of events in a queue. In this work a mathematical dynamical systems approach was used (1) to investigate the sensitivity of computer simulations to the effects of computer arithmetic on modeling computational processes and (2) to investigate the occurrence of chaotic behavior in mathematical models of attrition and reinforcement.

Sources in which a part of the material was published previously are:

1. "Dynamical Instability in Combat Models," *PHALANX—The Bulletin of Military Operations Research*, Vol. 25, No. 4, December 1992, pp 13, 24–26
2. "Dynamical Instability in Combat Models," in Proceedings of the 60th Military Operations Research Society Symposium, June 1992, pp 23–41
3. "Dynamical Instability in Combat Models: Computer Arithmetic and Mathematical Models of Attrition and Reinforcement," *Military Operations Research*, vol 2, no. 1, Spring 1996, pp 45–52.

Scope

This research should be regarded as a preliminary inquiry into the main problem of representing time and discrete events in combat simulations. The problem will not be solved until robust methods are used to join together simulations that employ a global time variable and simulations that have no global time, and are structured on the occurrence of discrete events. This problem represents an ongoing area of research in modeling and simulation.

2 Problems in Combat Modeling and Simulation

This chapter discusses significant problems inherent in the design of combat models and computer simulations:

- dynamical instability
- structural variance
- computer arithmetic
- the "base of sand."

Combat Models

Combat models—*nonlinear* deterministic models of decisionmaking processes that deal with attrition of opposing forces—contain dynamical instabilities. A combat model is designed using at least three different paradigms:

1. *discrete event simulations*
2. *dynamical systems*
3. *models of computer arithmetic.*

A model that uses several interacting modeling paradigms is a *complex model* [Palmore 1992d]. Discrete event simulations process event lists and do not have *global time variables*. Dynamical systems do. This is because dynamical systems incorporate laws of evolution that require global time variables. These laws are the algorithms used to compute the evolution of state variables. Numeric coprocessors provide high-speed computer arithmetic. They process the computations used by the iterative dynamics algorithms. Dynamical instability refers to divergences between solutions of dynamical systems. Dynamical instability destroys the robustness of a simulation and interferes with a simulation's *consistency*, reliability, transportability, and validity. Instabilities are caused by (1) computer arithmetic, (2) structurally unstable implementations of branchings on thresholds set by decision tables, and (3) structural defects in decision logic. Variability of computations results in differences in times of branchings on combinations of thresholds set by decision tables. Variability in branching sequences leads to *global divergence*. Unexplained global divergences of simulation

runs are often called "structural variance" in the military modeling community. Dynamical instability affects significantly verification, validation, and accreditation (VV&A) of models [Palmore 1992a]. The effects of dynamical instability on VV&A are studied to offer practical guidance for overcoming these obstacles [Palmore 1992b].

When is Structural Variance Important?

If the phenomena being modeled exhibit structural variance, then a functional requirement of the conceptual model is to exhibit structural variance to the extent shown by the activity being modeled [Davis 1992a].

If structural variance is unwanted—and assuming that the computer simulation has not been grossly misapplied—then there are several questions that may be asked about its appearance:

- Does an accepted computerized model of combat exhibit structural variance?
- Does it show structural variance in acceptable ranges of input data or in acceptable uses?
- Does structural variance occur in scenario domains of interest?
- Can structural variance be attributed to the conceptual model design or can it be isolated in the model's implementation as a computer simulation?
- To what extent is structural variance caused by models of computer arithmetic?

Interfaces in the Design of Combat Models

Design *interfaces* between an ambient discrete event simulation and its embedded dynamical systems have timing and synchronization problems. *Timing* refers to determining the time at which a threshold is crossed by a state variable as it is computed. *Synchronization* refers to making times of branchings correspond naturally to times on the event list. A fundamental problem with the use of dynamical systems in discrete event simulations is a lack of synchronization between the event sequence and threshold crossings by state variables. Threshold crossings enable branchings to occur. State variables of the embedded dynamical systems evolve by computation. The time at which a threshold is reached is not computed. Dynamical systems neither signal the discrete event simulation nor cause events to be created by it. This omission causes a mismatch between the two paradigms.

The problem is in the way a discrete event simulation updates its world state. This issue may be thought of as "viewing the world stroboscopically." Updating occurs only

after an event is processed. Only combinations of thresholds that have been reached previously are seen. There are no time tags signalling when thresholds are reached. Different computational procedures or different computer arithmetic can yield different states of the world and different branchings for sequences of events that correspond to threshold crossings of state variables.

A Stroboscopic View of the World

If a dynamical system is embedded in a discrete event simulation, then results of computation are viewed stroboscopically. Thus, decision tables are updated only as each event on the event list is processed. A stroboscopic view of a discrete event simulation's state is not synchronized with computed times when decisions are enabled.

A real-world example of a mismatch at an interface is given by a General Accounting Office (GAO) report on Patriot missile defense. Time is calculated in floating point arithmetic from data provided by an internal clock—an embedded dynamical system— in a discrete event decisionmaking program that decides when to launch a Patriot in defense. Similarly, attrition calculations in a combat model are run on an internal clock, the time variable of the attrition algorithm. Truncation in the 24-bit arithmetic caused errors to accumulate over a period of 100 hours. The range gate in the radar detection logic was shifted by this timing error and caused a failure to detect an incoming Scud missile [GAO 1992].

The problems created by this lack of synchronization can be more fully described as follows. One way is to make a threshold crossing an event and to make decisions when branching conditions are met by the newly created events. This increases enormously the numbers of events on event lists. Interfaces in time are created between a discrete event simulation and its embedded dynamical systems. This forces frequent checks of the world state in this enlarged set of events. It is not enough to synchronize the "stroboscope." The time at which a threshold is reached must be known in order to time-tag it. A second way is to implement the model as a dynamical system with a global time variable. This second choice is difficult to implement because many processes are hard to recognize as dynamical systems.

Determining When Decisions Are Enabled Dynamically

One needs to know when decisions are enabled. Decision times are times at which branching conditions are met. Branchings are enabled when combinations of thres-

holds are reached. The times when branchings are enabled can be added to times on the event list.

Computing algorithms do not measure times that the threshold is *reached*—they measure the first time at which a threshold has been *crossed*. The problem of determining when a threshold is crossed is a dynamics problem of long standing. Different algorithms yield different results. Estimating the time of threshold passage requires different algorithms. This need not be a serious problem in dynamical systems without branching, but when dynamical systems are coupled to decisionmaking where branching occurs, it is necessary to take precautions to ensure that branching is stabilized structurally. However, branchings depend on computer arithmetic.

Sources of Instability in Models and Simulations

Models of computer arithmetic affect the output of simulation runs. By changing algorithms that are used for computation of state variables one can affect the time at which branchings on thresholds set by decision tables occur. These changes, however small, may be sufficiently large to change the time at which threshold crossings are seen by the simulation. A much smaller effect, but of no less significance, is that caused by changes made in computer arithmetic. In this case divergences in the values of state variables may be observed.

Sensitivity of Simulations to Computer Arithmetic

Computer arithmetic is the implementation in *finite precision* of addition, multiplication, subtraction, and division. Fundamental register operations are shifts and adds. There are several choices of division algorithms. Computer arithmetic depends upon base of arithmetic, data type, precision, rounding, and specification. Institute of Electrical and Electronics Engineers (IEEE) standards for binary floating point arithmetic specifies 12 combinations of precision and rounding. The precisions are 24 (single), 53 (double), and 64 (extended). The rounding modes are round to nearest, and three directed rounding modes: truncate (chop or round toward 0), round up (toward $+\infty$), and round down (toward $-\infty$). Each combination of precision and rounding gives a different *model of computer arithmetic*. These 12 precision and rounding pairs comprise the basis of a "poor man's test" for model sensitivity. This test is recommended especially for models that have structural variance.

It is vital to recognize the importance of computer arithmetic and its vagaries in the execution of simulations of complex models where unstable behavior in embedded

dynamical systems—even the arithmetic itself—can cause excessive divergences in results [Palmore 1992d, 1991a, 1991b; Palmore and Herring 1990].

Chaos in Computer Arithmetic

The fundamental register operations of shift and add are implementations in hardware of a Bernoulli shift on the continuum. Bernoulli shifts are chaotic. Divergences grow rapidly. When exact answers are changed by rounding, the results of computing algorithms change slightly. Divergences between exact computations and results of computer arithmetic increase significantly when algorithms are sensitive to small changes. Pseudo random number generators are shifts implemented with integer arithmetic to provide exact results.

Effects of Computer Arithmetic on Iterative Processes

Computer arithmetic affects iterative computational processes. Time is measured by the number of iterations performed. Rounding and precision affect outcomes as iteration proceeds.

When are effects of computer arithmetic observed? The results of iterating discrete systems can be compared directly with evaluations of analytic solutions (if known). When an attractor of the dynamics exists the only effect likely to be observed is a change in phase of outcomes as the model of arithmetic is changed. There may be too few iterations for the effects of computer arithmetic to accumulate. It is a matter of how long it takes before significant phase changes between the outcomes occur using two different models of computer arithmetic. One measure of the effect is change in the win-loss picture of battle outcomes. How many steps will a change take? Figure 1 illustrates an effect of computer arithmetic. Different time signals are produced by rewriting a line of code to reorder arithmetic operations, or by changing algorithms in sensitive simulations [Palmore 1991a, 1992a].

Finding Bedrock Under the "Base of Sand"

Paul Davis describes a "base of sand" upon which, he argues, military combat modeling is founded [Davis and Blumenthal 1991]. Davis asks whether there is a science of military modeling or not. The author's work seeks an answer to Davis' question by finding the bedrock underlying the base of sand. Part of a solution depends on the awareness that the base of sand extends into a computer environment. Only by fully
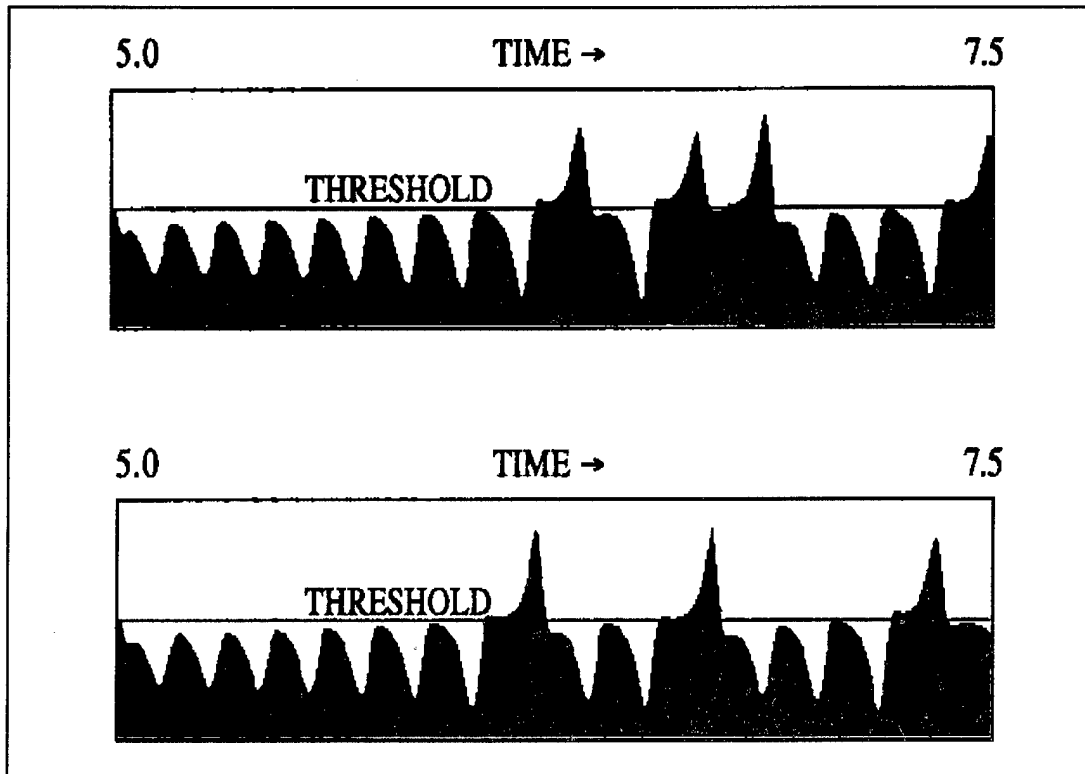
Figure 1. Altering time signals by changing computing procedures.

understanding the implications of using high-level languages in the design of models can the modeling community come to grips with fundamental issues of model design and implementation.

The base of sand extends into the realm of VV&A because goals of verification and validation are to demonstrate the degrees to which (1) a simulation performs as intended and (2) the real-world activity is modeled as intended. Meeting these goals requires knowing what the model does, observing how it does it, and showing that all model functional requirements are met. It seems clear that these goals cannot be accomplished without detailed knowledge of the influences of computer arithmetic, hidden assumptions in high-level languages, and knowledge of structural defects in the model and its simulation.

Summary of the Problems

Dynamical instability occurs in dynamical systems; structural variance occurs in discrete event simulations. Dynamical systems have laws of evolution for state variables and global time variables. When dynamical systems are embedded in discrete event simulations, dynamical instability can cause massive structural variance. If chaotic processes implemented for algorithmic computations are not embedded properly in

discrete event simulations, then dynamical instability causes structural variance. Table 1 suggests strategies for eliminating and reducing structural variance. Table 2 lists several principles of design. Table 3 lists *sensitivity analysis* considerations, and Table 4 identifies sources of instability in models and simulations.

Table 1. Strategies for eliminating or reducing structural variance.

1. Model dynamical processes as dynamical systems.

2. Model the dynamical systems with exact arithmetic. This produces robustness, consistency, and transportability.

3. Isolate the use of dynamical systems in models. Verify these independently. Make sure that the proper interface between the dynamical process and other architecture exists, and that the interface is exact.

4. Ensure that all branchings on thresholds are designed consistently for robustness and stability. If stroboscopic methods are used, then ensure that branching is based on exact arithmetic.

Table 2. Principles of model design.

Principle 1: Understand digital system architecture

Assumptions in digital system architecture affect the results of computer simulations. By understanding the system architecture, one has better opportunities for designing robust and stable simulations.

Principle 2: Use exact arithmetic and aggregate to simpler expressions

Perform exact arithmetic and aggregate the results when expressions become too large for practical use. This has two advantages. Results are exact before aggregation occurs. Differences are known when aggregation occurs.

Principle 3: Interfaces between paradigms must be designed properly

There are at least three paradigms in complex models of decisionmaking processes: (1) discrete event simulation, (2) embedded dynamical systems for computational purposes, and (3) models of computer arithmetic. Interfaces between an ambient discrete event simulation, its embedded dynamical systems, and computer arithmetic must be designed properly.

Table 3. Sensitivity analysis considerations.

1. Designing a well thought out experiment with well defined goals is necessary before making an attempt to conduct a sensitivity analysis. The question "What do we learn by a sensitivity analysis?" should be answered. Goals include measuring the sensitivities of the model and simulation. Is the model sensitive where intended? Is it overly sensitive?

2. Sensitivity analysis for computer simulations can be modeled on dynamical systems theory. Even though a discrete event simulation may not be written as a dynamical system, the embedded dynamical systems can be verified independently and their sensitivity examined.

3. Graphical representations using lattices of input data are alternatives to making long runs.

4. Global conclusions can be derived from a sensitivity analysis performed intensively throughout the data space. A local analysis usually does not imply global conclusions.

Table 4. Instabilities in models and simulations.

Causes

Causes

1. Models of computation, models of arithmetic, and computer arithmetic

2. Algorithms in models and their implementations in simulations

3. Branching on thresholds set by decision tables and their implementation in designs that use two paradigms with a mismatch in time at interfaces

4. Decision logic and its implementation in a high-level language that uses default settings that are not controlled by modelers

5. Structural instability of models and implementation in computer simulations by a structurally unstable methodology

Effects of instabilities on V & V

1. Unexplained sensitivity of models and simulations to initial data

2. Unexplained structural variances

3 Computer Arithmetic and Its Effects on Computations

This chapter discusses computer arithmetic, a contributing cause of instability in combat models and complex simulations. The paradigm of dynamics and dynamical systems is the setting for this discussion. One reason for this is that computations are performed by algorithms. If the computation requires algorithmic iteration, then the algorithm is a dynamical system that may have *computational chaos* in it [Palmore 1991a]. Complex models are decisionmaking models designed using two or more different interacting modeling paradigms [Palmore 1992d]. Combat models are complex models. Vagaries of computer arithmetic cause phase changes and, therefore, timing problems, in computational intensive discrete event simulations of decision-making models. For a discussion of the modern idea of paradigm, see Kuhn 1970. For a discussion of verification and validation of complex models see Palmore 1992d.

Discussions of computer arithmetic focus on instabilities inherited from real arithmetic. The section that follows is an analysis of computer arithmetic effects for the Patriot example noted in Chapter 2. Predicting the extent of divergence introduced by vagaries of computer arithmetic in complex simulations is difficult because of the enormous numbers of computational paths. In the Patriot example, however, there is only one computational path of significance.

A Detailed Analysis of the Patriot Example

A timing problem is illustrated by a significant real-world example in which an attack by a Scud missile on Dhahran, Saudi Arabia, during the Persian Gulf War, was not engaged. As documented in a GAO report, an accumulation of timing error caused the failure of a radar system to detect and to identify a missile that should have been defended against by the Patriot [GAO 1992]. This, in turn, resulted in a failure to launch a Patriot. The timing problem was a mismatch between a timing computation and the stroboscopic view by the radar. Time was accumulated as integer numbers of tenths of a second. Thus, 100 tenths equals 10 seconds. However, the arithmetic was done in binary, not in decimal. If finite precision is used, there is always a rounding error in the binary representation of 1/10. The timing error that results by accumulation from the truncated expansion of 1/10 is directly proportional to the integer number of

tenths. When the system operates for more than a few hours, this error is appreciable. In this particular case, the system operated for 100 hours or 3,600,000 tenths of a second. Computer arithmetic with 24 bits was used for floating point calculations. A 24-bit word used to represent a number contains a sign bit, an exponent, and a significand. Precision refers to the number of bits in a significand (or fractional part). Precision 20 results in an error of about 2^{-23} in the binary representation of 1/10. One tenth is written as a binary expansion 0.00011001100110011001100|1100... where truncation error is about 2^{-23} at precision 20. The difference between 1/10 and the expansion truncated at precision 20 is exactly 0.8×2^{-23}. The floating point calculation is off by $3,600,000 \times 0.8 \times 2^{-23}$ (= 0.343322...), about 1/3 second. This error caused a shift of about 700 meters in the range gate (= 1/3 sec × 2000 m/s, the terminal velocity of Scud), which was sufficient for the radar detection logic to fail to identify and flag the incoming Scud as a missile that should be engaged [Palmore 1992a]. See Table 5.

The Job of Computer Arithmetic

Computer arithmetic is the implementation in hardware of finite precision real arithmetic. Real arithmetic consists of the field of real numbers with the binary operations of addition and multiplication (+, ×). The field axioms are (1) associativity and commutativity in addition and multiplication, (2) distributivity with respect to addition and multiplication, (3) the existence of additive and multiplicative inverses, and (4) identities for addition and multiplication. *Multiplication is iterative addition of two real numbers* and *division is iterative subtraction of a nonzero real number from a real number.* These operations need not terminate in a finite number of steps for arbitrary computable real numbers because arbitrary computable reals have infinite

Table 5. Effects of extended run time on Patriot Operation.

HOURS	SECONDS[†]	CALCULATED TIME (SECONDS)	INACCURACY (SECONDS)	SHIFT (METERS)
0	0	0	0	0
1	3600	3599.9966	.0034	7
8	28800	28799.9725	.0275	55
20	72000	71999.9313	.0687	137
48	172800	172799.8352	.1648	330
72	259200	259199.7528	.2472	494
100	360000	* 359999.6667	.3433	687

Source: GAO/IMTEC-92-26, Patriot Missile Software Problem, Appendix II.
[†] This column is calculated exactly as 10 × Integer count of 1/10 seconds.
* The digit **6**, represented in boldface type, is a misprint in the GAO report.

arbitrary computable real numbers because arbitrary computable reals have infinite digit strings. Instabilities of real arithmetic are inherited by finite precision computer arithmetic.

A division computation diverges rapidly once a mistake is made. Division is unstable because division is equivalent to a Bernoulli shift, a known chaotic operation [Palmore and McCauley 1987; Palmore 1988; Palmore and Herring 1990].

Models of Computer Arithmetic

Models of computer arithmetic include algorithms used to implement the operations of addition, multiplication, subtraction, and division. These algorithms for arithmetic are implemented usually in hardware within the numeric coprocessor. Although the fundamental elements of shift and add are employed, there is a choice in the algorithm used for division. Models of computer arithmetic depend upon base of arithmetic, data type, precision, rounding, and specification.

The IEEE standards for binary floating point arithmetic and radix independent floating point arithmetic, noted in Chapter 2, allow 12 combinations of precision and rounding [ANSI/IEEE 1985]. IEEE precisions are 24, 53, and 64 binary bits. These are single, double, and extended precisions, respectively. IEEE rounding modes are round to nearest, and three directed rounding modes: truncate (chop or round toward 0), round up (toward $+\infty$), round down (toward $-\infty$).

Each combination of precision and rounding is a different model of computer arithmetic for the computation of numerical quantities. The 12 combinations of precision and rounding allow a "poor man's test" of sensitivity. By running a computer simulation using different combinations of precision and rounding, a designer can test directly its sensitivity to computer arithmetic.

These models of computer arithmetic make a difference. For example, when using a pseudo random number generator (PRNG) of the form $x \rightarrow ax \bmod m$ (a *prime modulus multiplicative linear congruential generator*) with $a = 7^5 (= 16807)$ and $m = 2^{31} - 1 (= 2147483647)$ at least 46 bits of precision are needed to contain intermediate numerical results. Consistent answers cannot be obtained if this requirement is not met. The reason for this is that when rounding occurs in the integer arithmetic (in the floating point environment where less than 46 bits are allowed), the multiplying factor of 16807 causes immediate divergence between the exact PRNG sequence and the computed sequence [Palmore 1992b, 1991a,b].

Consider the Simscript II.5 PRNG. The multiplier a = 630360016 and prime m = 2^{31} - 1 are the parameters [Fishman and Moore 1986]. Intermediate products of multiplication exceed double precision. The code uses partial products

$$(A2^N + B)\cdot(C2^N + D)=AC\cdot2^{2N} + (AD + BC)\cdot2^N + BD.$$

The Simscript II.5 generator fits existing double precision by computing and retaining partial products AC, AD, BC, and BD. Special care has been taken to implement this simple PRNG so that there is 1 to 1 correspondence between the discrete arithmetic model and its computer implementation.

Effects of Tiny Roundings on Simulations

If computer simulations are computational-intensive, then the computations—performed typically by algorithms as iterative operations—can inherit instabilities from computer arithmetic. The important point is this: when algorithms perform calculations efficiently, a lot of information is gained at each step. For division at least one digit is gained at every step. Very efficient division algorithms, such as Newton's method, allow a geometrically increasing number of digits to be gained step by step. Chaotic processes are being used for this purpose. If small errors are found in an unstable calculation, then a rapid divergence of solutions is guaranteed.

A Cure for The Propagation of Errors

A cure for the propagation of errors is to use indefinite precision calculations so that small errors remain small. When a calculation hits a boundary, an effect arises that is similar to turbulence in fluids. In such a case, finite precision establishes a "boundary layer" in which turbulence is produced and extends to chaos in the "bulk calculation."

When indefinite precision is used, the number of bits in a calculation is allowed to grow to a much larger number than standard precisions allow. Bulk effects are seen only when a boundary is reached and back propagation occurs. This approach is effective when an attempt is made to hold accuracy to a few digits uniformly throughout a calculation. Thus, one ensures enough room to work so that the number of steps available for errors to propagate far enough in precision to reach a boundary, to be reflected, and to propagate back to affect the accuracy desired, is greater than the number of steps required to complete the calculation.

Availability of Indefinite Precision Arithmetic for Exact Computations

Indefinite precision is not available in high-level discrete event simulation languages. Indefinite precision has to be built especially for particular calculations. It is part of the design of algorithms and their implementations on machines.

Computational programs are available to do indefinite precision arithmetic, including rational arithmetic where answers are retained as rationals. These programs give exact answers.

Information Loss in Computer Arithmetic

Information is lost in finite precision arithmetic. Consider the set of significands in precision N that represent numbers in the interval $1 \leq x < 2$. What is the subset of significands of numbers such that the reciprocal of each, expanded in base 2, has period not exceeding N? Each rational number has a binary expansion that is eventually periodic. Most of the binary expansions of reciprocals have periods that exceed N. Thus, finite precision does not hold the information necessary to represent the period of the binary expansions of most numbers. These defects may appear to be highly technical in origin. They are fundamental in the sense that changes in precision and rounding do not change their effects.

Effects of Computer Arithmetic on Iterative Processes

Problems with computer arithmetic can arise in iterative processes such as discrete dynamical systems. An example is given by the difference equations that are written down from the continuous system of Lanchestrian differential equations. These differential equations have hyperbolas as solution curves. A significant point is to model the reinforcement strategy easily. As in every discrete dynamical system in which the model is a set of difference equations to be iterated with uniform step size, the time is measured by the number of iterations performed. In particular, roundings and precisions will affect outcomes as iteration proceeds.

Here is the main question: when will the effects of computer arithmetic be observed? The results of iterating the discrete system can be compared directly with evaluations of the analytic solution. When an attractor of the dynamics exists the only effect likely to be observed is a change in phase of outcomes as the model of arithmetic is changed. With limited numbers of reinforcement blocks, there may be too little time (that is, too few steps) for computer arithmetic effects to accumulate. For chaotic behavior, with

unlimited numbers of reinforcement blocks and unlimited numbers of iterations, it is a question of how long it takes before significant phase changes between the outcomes occur using two different models of arithmetic. One measure of effect is changing the win-loss picture in a combat simulation. How many steps will this take [Palmore 1991a, c]?

4 Combat Models and Structural Variance

A Definition of Combat Models

A complex model is a model of decisionmaking processes that is designed using several different interacting modeling paradigms [Palmore 1992d]. Combat models are examples of complex models. In particular, a combat model is a model of decision-making processes that deal with attrition of opposing forces. It is implemented as a discrete event simulation with embedded dynamical systems. Any discrete event simulation that depends on floating point computation is a complex model. The reason for this is that computer arithmetic itself satisfies the properties of an embedded dynamical system. We emphasize the dependence of discrete event simulations on embedded dynamical systems in which intensive computation is done. The difference between the paradigm of the discrete event simulation and that of the dynamical system is that the dynamical system has a uniform time base and the discrete event simulation does not. This may appear to be of no consequence, but this difference in paradigms has the great consequence of rendering the discrete event simulation with the problem of structural instability that results in structural variance of outcome.

Synchronization

Synchronization is important. *Resolution* of simultaneous events has been studied for many years. The intercept problem—that of determining how to intercept a target—has been studied as a control problem in mathematics. It poses interesting mathematical questions. In celestial mechanics, the intercept problem takes a form of planetary flyby missions where the goal is to use minimal energy to pass by several planets. Control is required to adjust velocities in midcourse so that a desired trajectory is followed. Synchronization of events is a difficult and fundamental problem that is well known historically in the computational sciences.

Synchronization of branching depends upon knowing the time at which an event occurs. A discrete event simulation looks at the state space when an event occurs. The simulation updates the "state of the world" at that time. A problem arises because the simulation makes no provision for tagging as an event the time at which a threshold is reached. This leads to excessive divergences in output.

Combat Models Revisited

As noted, a combat model has dynamical systems embedded in its discrete event simulation. Dynamical systems compute state variables so that branching on thresholds of values set by decision tables can occur. Thus, there is a mismatch of the two paradigms: the discrete event simulation has no "continuous" time line, but each dynamical system has its own uniform time line. An event occurs and decision tables are checked by the discrete event simulation. Decisions are made at that time based on the values of state variables. If a threshold has been exceeded, then the lookup time is usually later than the time at which the threshold crossing occurred. Differences in computation can affect the times at which thresholds are crossed.

Slight phase differences in results of computations between two different machines may cause results to be on opposite sides of a threshold when a lookup occurs. Consequently, machine A sees that a threshold has been exceeded when event E occurs and machine B sees no threshold crossing when event E occurs. This stroboscopic viewing of dynamical systems surely defeats the goals of obtaining consistency, robustness, stability, and transportability of the simulation. It subjects the simulation to vagaries of computations and computer arithmetic. Consider an example of branching on thresholds set by decision tables where the checks of state variables are made stroboscopically. A simple example illustrates the effects produced in a complex model by using as a dynamical system a PRNG with a branching condition. If the *orbit of the dynamical system* falls in a segment of the interval, then the state variable is increased in value by 1 and the iteration continues. This is a dynamical system that is checked algorithmically at every step to determine whether the branching condition has been met or not.

Interfaces in Combat Model Design

A fundamental problem with the use of dynamical systems in discrete event simulations is to provide synchronization between the event sequence and crossing of thresholds by state variables within the dynamical systems. Thresholds in decision tables are used to trigger branching. State variables of the embedded dynamical systems are evolved by computation. When values of state variables exceed thresholds in decision tables and certain combinations of thresholds are crossed to enable decisions to be made, the stage is set for a mismatch between the two processes. The mismatch occurs because the time at which a threshold is crossed is not computed and neither signals to nor cause an event to be created by the discrete event simulation.

The problem with this is the way the discrete event simulation updates its state of the world. Updating occurs when an event is executed. This means that the state of the world is viewed only at times of events. Thus, the method picks out only those thresholds and combinations of thresholds that have been met for some time. There are no time tags to record when thresholds were met. Thus, different computational procedures and different arithmetic are guaranteed to yield different states of the world—and, consequently, different branchings—for those sequences of events that coincidentally correspond with threshold crossings of state variables. This is not a rare occurrence.

There is a solution to this difficulty, but the solution presents another fundamental difficulty. If the times at which thresholds were crossed were computed, then these crossings could be labelled by the simulation as events. However, this determination by calculation requires a separate approach. The problem of determining when a threshold is crossed or the time at which a state variable takes a particular value is difficult to resolve. A reason for this difficulty is that algorithms implemented by machine are necessarily discrete and have appropriate step sizes. By checking values of state variables it is clear that one can determine the fact that a threshold value has been exceeded.

Thus, if one has two times of successive steps of the algorithm that determines S, t_1 and t_2, such that at t_1, $S(t_1) < M$ and at t_2, $S(t_2) > M$, it may be possible to estimate an intermediate time, t^*, at which $S(t^*) = M$. The problem remains to estimate the intermediate time to within a prescribed accuracy in order to determine the ordering of t^* and $t_A < t_B$, which are the times of events A and B. The reason for this is that an ordering $t_1 < t_A < t^* < t_B < t_2$ is possible. Thus, the state of the world at neither t_A or t_B includes the threshold crossing which is first observed at t_2. Different step sizes yield different outcomes and affect the simulation in different ways. The problem of determining to within a prescribed accuracy the time at which a threshold is crossed is a dynamics problem of long standing. Different algorithms yield different results. Even estimating the time of threshold passage requires the use of a different algorithm. In dynamical systems without branching this need not be a serious problem. But when dynamical systems are coupled to discrete event simulations where branching occurs, it is necessary to take strict precautions in order to ensure that branching is stabilized structurally.

Structural Variance

Where does structural variance in complex models come from? It arises from instabilities in the model's embedded dynamical systems that are amplified by

decisionmaking processes such as branching on thresholds set by decision tables. There is a fundamental difficulty in synchronizing events triggered by dynamical systems that are embedded in discrete event simulations. A particular problem is to design a method for synchronizing branching on thresholds of state variables where branching is controlled by decision tables in a discrete event simulation and the evolution of state variables is computed within the embedded dynamical systems.

The fundamental difficulty is to determine the time at which thresholds are reached. This requires algorithmic solutions, even for differential equations, because either inversion of analytic solutions or iteration of numerical methods is required to find an approximate solution for the time that a threshold is met. The discussion that follows addresses this problem and cases where the solution is known, as well as possible solution strategies for the general problem. See also the discussion of this problem in Palmore 1991a.

Sensitivity Analysis

Sensitivity analysis plays a key role in verifying a computer simulation. Measuring the degree of sensitivity of a model to various factors is a goal of any sensitivity analysis. Additionally, sensitivity analysis should provide information about the response of branching and decision logic to changes in initial data, thresholds, and methods of computing state variables. Branching on thresholds of state variables set by decision tables requires computing the state variables and recognizing that thresholds have been exceeded. Part of the sensitivity of branching is caused by the discrete event simulation paradigm that looks at the state of the simulation stroboscopically. Because there is no continuous time base for the discrete event simulation, this paradigm does not identify the times at which thresholds are crossed. This leads to a mismatch between the dynamical systems behavior and that of the ambient simulation.

Several questions should be answered by a sensitivity analysis. Is the model/simulation sensitive where intended? Are there unwanted sensitivities in the model/ simulation? How can sensors be placed effectively so sensitivities can be measured? Can the sensitivity of the model/simulation be increased or decreased? What is the best way to test for sensitivities after the model is completed? How can sensitivity analysis help a VV&A process?

Scenario Development

Scenario development is an iterative process that affords a measure of the sensitivity of a computer model and simulation. A scenario for a model is designed iteratively by successively starting the model with an initial data set at time t=0 and running the model until difficulties are encountered. The difficulties may be bugs in the code, crashes, divergences, unexplained and unwanted structural variance, or other anomalies. Divergences may be observed at a particular time $t = t_1$.

The run is continued for time in excess of t_1 to observe any continuation of the behavior. Then at time $t_2 > t_1$, when further evaluation is unnecessary, the run is stopped.

The initial data set is reconfigured. The model is started again at time $t = 0$. The run continues until it crashes or divergences appear and the data set is reconfigured. This iterative design process continues until runs are free from apparent difficulties for the length of run time desired. Obviously, this process is one of avoiding critical paths both locally and globally. How successful is this process? There is no guarantee that the iterative process will converge satisfactorily with respect to given stopping criteria.

What criteria are used to stop the design process? If the data set were designed by automatic means, then convergence is guaranteed because one stays within a basin of attraction of a fixed point for the method. Otherwise, in the case of hand-implemented design processes, where stopping criteria may change from step to step, there is no guarantee that an iterative design process converges. For example, if the designers move from one basin of attraction of the automated process to another as changes are made, then the result will be an oscillation of the design between basins of attraction after several steps. In this case, satisfactory data sets will take a long time to develop—perhaps as long as several months.

5 Dynamical Systems Theory

Dynamics deals with systems that evolve with time. Dynamical systems may be either continuous time dynamical systems or discrete time dynamical systems. Continuous time dynamical systems arise as solution sets of differential equations. Discrete time dynamical systems are mappings of a space into itself. The state space or domain of the dynamical system may be continuous or discrete. Table 6 shows several types and choices available. It should be noted that for every set of choices of dynamical system classification there is a choice of domain.

A dynamical system is defined here as a family of maps that arises from continuous time or discrete time dynamics. Let X be a state space and let I denote a time domain. Let Φ be a map $\Phi: I \times X \to X$. The time domain I is usually a connected subset of the real numbers or a set of successive integers. We assume that I is a nonempty symmetric set so that if $t \in I$, then $-t \in I$. In particular, $t = 0$ is in I. The state space X may be a collection of random variables, Euclidean space, or a *function space*. For all $t \in I$ and for all $x \in X$, we define a family of maps $\{\Phi_t\}_{t \in I}$ by the rule $\Phi_t(x) = \Phi(t,x)$. The map Φ has the following two properties:

Table 6. Classifying dynamical systems.

CHARACTERISTICS OF DYNAMICAL SYSTEMS

deterministic	or	stochastic
discrete	or	continuous
unique	or	nonunique
robust	or	weak
structurally stable	or	structurally unstable
regular	or	chaotic
monotonic	or	nonmonotonic

DOMAINS OF DYNAMICAL SYSTEMS

TIME	*and*	*SPACE*
continuous	and	continuous
continuous	and	discrete
discrete	and	continuous
discrete	and	discrete

1. For all $x \in X$, $\Phi(0,x) = x$; that is, Φ_0 is the identity on X
2. For all $t, s \in I$ such that $t+s \in I$, and $\Phi_t(\Phi_s) = \Phi_{t+s} = \Phi_s(\Phi_t)$; thus, $\Phi(t,\Phi(s,x)) = \Phi(t+s,x)$ for all $x \in X$.

For $t \geq 0$, an inverse is defined by $\Phi_t(\Phi_{-t}) = \Phi_{t-t} = \Phi_0$ or $\Phi(t,\Phi(-t,x)) = \Phi(t-t,\Phi(0,x)) = x$ for all $x \in X$. Examples of dynamical systems are given by (1) a smooth map f: $X \to X$, (2) a smooth diffeomorphism f with smooth inverse g, (3) a smooth semiflow $\{\Phi_t\}_{t>0}$, and 4) a smooth flow $\{\Phi_t\}_{t \in R}$. Here "smooth" means of class C^r for an $r \geq 1$.

Although f may not be invertible, a generalized inverse may exist as a multivalued function. In this case, "f^{-1}" has several branches $\{f_i^{-1}\}$ that are the inverses to a decomposition of f into $\{f_1, ..., f_n\}$. This means that the history of the solution is not lost if a record of each branching is made.

There are two properties of solutions of any dynamical system:

1. Each solution $\phi(t,x_0)$ satisfies $\phi(t+s,x_0)=\phi(t,\phi(s,x_0))$, $t,s \geq 0$
2. Each solution $\phi(t,x_0)$ has an inverse $\phi(-t,\phi(t,x_0)) = x_0$, $t \geq 0$.

These imply that a model's law of evolution should be written for both time directions. By implementing into a computer simulation each direction of evolution the simulation is designed so it can be run in reverse. This provides independent 1-to-1 correspondences between the model and its computer implementations. An example of this process is implementing PRNGs and their inverses, even with branching on thresholds [Palmore 1991b]. A successful implementation results from a complete verification of the computation in the design and simulation.

Computer Simulations

Properties inherent in a dynamical system imply the following for computer simulations: (1) the computer simulation must be capable of being stopped at time t^* and restarted from time t^* to reach the goal as is reached without stopping the run, and (2) the computer simulation should be capable of being run in reverse. These conditions have implications for verifying and validating models and computer simulations:

1. A computer simulation should support stopping at time t>0 and restarting at time t>0. The semigroup property of composition is inherent in all definitions of dynamical systems. This action is not provided for by most simulations of deterministic *dynamical processes*. To the extent that the ideal is not met, the computer simulation fails to be a valid implementation of a dynamical system.
2. A computer simulation should be reversible. Invertibility is a property of mathematically formulated dynamics. A dynamical model that is implemented as an invertible simulation in both time directions is verifiable.

There is an interesting restriction that appears after stating a definition of dynamical systems: computer simulations are usually not *written* in the paradigm of dynamical systems, but computer simulations *contain* dynamical systems. There is a distinction.

In the discrete event simulation, values of time are $t \geq t_0$ (the initial time, usually $t_0 = 0$). Although no history may exist prior to t_0, it is important to be able to reverse a simulation from time $t > t_0$ to run backward to the initial time.

Chaotic Dynamical Systems

The study of chaos shows that chaotic processes on the continuum can be restricted to a computable setting. For example, division of a real number by a nonzero real number can be accomplished by shifts and subtractions and, therefore, it can be viewed as iterative subtraction. Taking the reciprocal of an integer produces an expansion that is eventually periodic in any base of arithmetic. Multiplication is a process of iterative addition. Multiplication of infinite expansions of two reciprocals as real numbers is a process that does not terminate. Arithmetic operations of real arithmetic, when viewed in an infinite precision computable setting, exhibit all of the chaos properties of operations on the continuum.

A Bernoulli shift of the form: $X \rightarrow 2X \bmod 1$, $0 \leq X < 1$, is a chaotic dynamical system. This is the most elementary chaotic system on the interval. It is piecewise linear and has a Lyapunov multiplier of 2. The Lyapunov exponent is ln 2. For a number X that is written in binary, the Bernoulli shift takes the following elemental form:

$$X = 0.a_1 a_2 ... \rightarrow 2X \bmod 1 = 0.a_2 a_3 ... \,.$$

It is represented obviously by a shift of the base point one position to the right followed by a truncation of the bit to the left of the repositioned base point. Thus, if we decompose this operation visually, we obtain the steps:

$$X = 0.a_1 a_2 a_3 ... \rightarrow 2X = a_1 . a_2 a_3 ... \rightarrow 2X \bmod 1 = 0.a_2 a_3 ...$$

The Bernoulli shift is the mapping used for PRNGs. In a prime modulus multiplicative linear congruential generator (PMMLCG) the shift takes the form: $X \rightarrow AX \bmod M$, where X is an integer in the range $0 \leq X < M$, modulus M is prime, and multiplier A is chosen so there is only one cycle in addition to the fixed point of the shift $X = 0$. The cycle has length $M - 1$.

A typical choice for (A,M) is $A = 16807 = 7^5$ and $M = 2^{31} - 1 = 2147483647$, a well known prime number. Because the PMMLCGs use prime modulus, one has a finite field. This allows an inverse PMMLCG to exist so that the PRNG can be run forward or backward.

The implementation of the PMMLCG design in a computer arithmetic environment is verified completely because a 1-to-1 correspondence between the discrete mathematical model and the simulation in arithmetic has been established. Clearly, a chaotic mapping on the continuum yields a very practical design for a PRNG.

Disordering by Chaotic Dynamics

What are the properties of Bernoulli shifts inherited by a pseudorandom number generator? What are the properties that make them so attractive for this purpose? How can a deterministic device be used to generate a sequence of numbers that is in turn used to implement random number sequences? A Bernoulli shift on binary bit strings has the following property: for almost every number x, $0 \le x < 1$, the orbit of x is uniformly distributed. Almost every number in the interval is a normal number. Computable numbers are a set of measure zero.

Bernoulli shifts are the simplest to implement in code as PRNGs. The program has a complexity on the order of the logarithm of the prime modulus. Thus, Bernoulli shifts on a lattice are not algorithmically complex. However, Bernoulli shifts disorder sets of integers and do so rapidly. This disordering effect of the PMMLCG is due to stretching and wraparound. The multiplier stretches and the modular operation is the result of wraparound.

The deterministic PMMLCG generates a stream of numbers that carry an apparent maximal disorder. This is the way in which the generator mimics randomness. However, what is produced has no randomness associated with it. All that is given is a disordered stream of integers. Thus, the purpose of the pseudorandom number generator is to disorder a set of ordered integers.

A Criterion for Mathematical Chaos

For differentiable maps on continuous domains a criterion for chaos is expansion and wraparound or folding of motion. For domains with compact support, expansion of motion implies wraparound or folding. Expansion and contraction are measured by derivatives. Expansion along an orbit requires that *on the average* there is at least one eigenvalue with magnitude greater than unity at points that are visited by a chaotic orbit. The phrase "on the average" is used here with a very precise mathematical meaning. It means intuitively that an eigenvalue has a magnitude greater than unity if its average value over all points on an orbit exceeds 1. However, for current

purposes a mathematical elaboration is not necessary. This is because the derivatives that are considered for simple models of attrition and reinforcement are constants.

If expansion holds throughout the domain of a map, then nonperiodic motion is chaotic. This criterion is a requirement that there be a Lyapunov multiplier Λ such that $ln|\Lambda|$ > 0. Lyapunov multipliers measure expansion and contraction of the motion. The natural logarithm of $|\Lambda|$ is the Lyapunov exponent. A positive exponent means that motion is expanding in a given direction.

Consider the logistic map on the interval, $0 \le x \le 1$, with parameter value equal to 4. The map is $f(x) = 4x(1-x)$. The derivative $f'(x) = 4 - 8x$. Thus, the magnitude of the derivative of f, $|f'|$, exceeds 1 for all x, $0 \le x < 3/8$, $5/8 < x \le 1$. However, along every orbit that is not periodic, the average value of $|f'|$ equals 2. This is the value of the Lyapunov multiplier; the Lyapunov exponent equals ln 2. The motion along a chaotic orbit is expanding *on the average* by a factor of 2.

For a piecewise linear map, this criterion for chaos is a condition on the eigenvalues of the map averaged over the orbit considered. In particular, for a map with constant derivative, the eigenvalues are constant and are the Lyapunov multipliers of the dynamics.

Stability Considerations

For continuous or discrete dynamical systems that are defined on continuous state spaces, the concept of stability of a solution is defined by divergence of nearby solutions. A solution $\phi(t)$, $t \ge 0$, is stable if given $\epsilon > 0$, there is a $\delta > 0$ such that every solution $\sigma(t)$ that starts within δ of $\phi(0)$ remains within ϵ of $\phi(t)$ for all $t \ge 0$. Thus, $|\phi(0) - \sigma(0)| < \delta$ implies $|\phi(t) - \sigma(t)| < \epsilon$, for all $t \ge 0$. The requirement "$t \ge 0$" means that t is a nonnegative integer for discrete systems and t is a nonnegative real number for continuous systems. Implied by the definition is the existence of nearby solutions for all $t \ge 0$.

A solution $\phi(t)$, $t \ge 0$, is unstable if it is not stable. Thus, $\phi(t)$, $t \ge 0$, is unstable if there is an $\epsilon > 0$ such that for all $\delta > 0$, $\delta \le \epsilon$, there is a solution $\sigma(t)$, $t \ge 0$, and a time $T > 0$, such that $|\phi(0) - \sigma(0)| < \delta$ and $|\phi(T) - \sigma(T)| > \epsilon$.

When dynamical systems are restricted to lattices the concepts of stability and instability take on different meanings because there are no solutions nearby. An orbit on a lattice that is vacuously stable may be unstable when embedded in a continuous domain. Thus, PRNG defined on an integer lattice is stable in a vacuous sense because

there are no nearby orbits. Thus, given $\epsilon > 0$ there is a $\delta > 0$ (any number sufficiently small will work) such that every orbit that starts within δ of (there are none) remains within ϵ of the given orbit. However, the PRNG is a Bernoulli shift that is restricted to a domain of integers. When this integer lattice is placed on the continuum, the orbit of the PMMLCG is unstable.

6 Chaos, Nonlinearity, and Nonmonotonicity in Simple Models of Attrition and Reinforcement

The interest in chaos and nonmonotonicity of simulated battle outcomes in simple models of attrition and reinforcement has come primarily from the work of Dewar, Gillogly, and Juncosa at RAND [Dewar et al. 1991]. A mathematical analysis of a generic model of attrition and reinforcement by the author demonstrates chaos as a consequence of a positive Lyapunov exponent and disorder given by moduli operations that simulate reinforcement [Palmore 1992a]. Unpredictability in actual battle outcomes is suggested by Singleton's work [Palmore 1992b].

The text that follows demonstrates how nonmonotonicity in battle outcomes is obtained for free by the addition of "area fire" terms to linear Lanchester equations. Chaos is proved to exist both in the linear discrete Lanchester equations with reinforcement and in the nonlinear discrete Lanchester equations with reinforcement.

Chaos and Simple Models of Attrition and Reinforcement

Consider the following system of first order linear differential equations of Lanchester:

$$dx/dt = - ay$$
$$dy/dt = - bx \qquad [1]$$

where x, y > 0, and auxiliary conditions

a.   if $y/x \geq r$, then x is replaced by x + c, repeatedly $N \geq 1$ times, until $y/(x+Nc) < r$ holds

b.   if $y/x \leq s$, then y is replaced by y + d, repeatedly $K \geq 1$ times, until $(y+Kd)/x > s$ holds, where a,b,c,d,r,s > 0, and $r > s > \sqrt{(b/a)}$.

This model has area preserving flow because the divergence of the right hand side equals 0. It has a first integral $I(x,y) = ay^2 - bx^2$. The solution curves of the

differential equations are hyperbolas and the solutions of the system of equations together with the auxiliary conditions are arcs of hyperbolas and isolated points.

A Discrete Model Derived From the Linear Model

By writing down a set of first order linear difference equations to replace the system of first order linear differential equations [1] above, a discrete time dynamical systems model of attrition and reinforcement is obtained. The system of linear difference equations is

$$x_{n+1} = x_n - hay_n$$
$$y_{n+1} = y_n - hbx_n \qquad [2]$$

where $x_n, y_n > 0$, for all $n \geq 0$, $h > 0$ is a parameter, and auxiliary conditions

a. if $y_n/x_n \geq r$, then x_n is replaced by $x_n + c$, repeatedly $N \geq 1$ times, until $y_n/(x_n+Nc) < r$ holds, and

b. if $y_n/x_n \leq s$, then y_n is replaced by $y_n + d$, repeatedly $K \geq 1$ times, until $(y_n+Kd)/x_n > s$ holds, where a,b,c,d,r,s, and $h > 0$, and $r > s > \sqrt{(b/a)}$.

For $h \neq 0$, the flow of the discrete model is *area decreasing* (the determinant of the mapping equals $1 - h^2ab < 1$). Only those values of h for which the determinant is positive, i.e., $1-h^2ab > 0$ are considered. An attractor of the discrete flow is a compact, invariant set. Therefore, any attractor has measure 0.

Lyapunov Exponents and a First Integral

Lyapunov multipliers and Lyapunov exponents are computed easily. The multipliers are given by $\Lambda_{\pm}(h) = 1 \pm h\sqrt{(ab)}$. Lyapunov exponents are $\lambda_{\pm}(h) = ln\ [1{\pm}h\sqrt{(ab)}]$. By scaling arguments for piecewise linear maps, the Hausdorff dimension of the attractor $A(h)$ can be computed as

$$HD[A(h)] = 1+\lambda_+(h)/|\lambda_-(h)|$$

for any $h > 0$. For $h \geq 0$, we have $HD[A(h)] = 2 - h\sqrt{(ab)} + O(h^2ab)$. Thus, the Hausdorff dimension of $A(h)$ is less than 2 for $h > 0$ so that A is a fractal attractor.

We have proved the existence of a fractal attractor, a strange attractor of the discrete time dynamics for every $h > 0$ such that the determinant of the mapping is positive.

A first integral for this discrete flow is given by $I(x_{n+1}, y_{n+1}; x_n, y_n) = (ay_{n+1}^2 - bx_{n+1}^2)/(ay_n^2 - bx_n^2) = 1-h^2ab$. This expression is independent of the orbit. The integral $I(x_{n+1}, y_{n+1}; x_n, y_n)$ is called an isoenergetic first integral and it depends only upon h, a, and b.

It is clear that chaos exists in this dynamical system: there is a strange attractor and the system has a positive Lyapunov exponent for every h > 0 considered. These conditions suffice to indicate the presence of chaos [Palmore 1991]. The positive Lyapunov exponent ensures a sensitive dependence on initial conditions for this discrete dynamical system. Chaos exists for all choices of h, a, and b such that $1-h^2ab$ > 0. In particular, the Hausdorff dimension of the attractor, HD[A], can be computed easily as a function of the parameters h, a, and b.

The cases considered by Dewar, Gillogly, and Juncosa of RAND are covered by the family of discrete dynamical systems above [Dewar et al. 1991]. In RAND's simple model of attrition and reinforcement the discrete dynamical system is piecewise linear with a constant derivative. From the analysis above several properties of the dynamics are immediately known: the Lyapunov multipliers, a positive Lyapunov exponent, the directions of expansion and contraction, the measure and dimension of the attractor. The dynamical system is chaotic by a positive Lyapunov exponent. This simple model of attrition and reinforcement is an example of predator-predator interaction with restocking of each predator. Combat is distinguished from other model types such as predator-prey, competing species, and logistic.

Nonmonotonicity and Nonlinearity in Models of Attrition and Reinforcement

The terms for "area fire" used by Lanchester can be added to the model of linear differential equations. The differential equations are written

$$dx/dt = - Ay - ACxy$$
$$dy/dt = - Bx - BDxy \qquad\qquad [3]$$

where x, y > 0, and auxiliary conditions

a. if $y/x \geq r$, then x is replaced by x + c, repeatedly N ≥ 1 times, until $y/(x+Nc) < r$ holds

b. if $y/x \leq s$, then y is replaced by y + d, repeatedly K ≥ 1 times, until $(y+Kd)/x > s$ holds, where A,B,C,D,c,d,r,s > 0, and $r > s > \sqrt{(B/A)}$.

The criterion $\sqrt{(B/A)}$ follows from analysis of the integral curve as it approaches the origin. This model has area decreasing flow (the divergence of the right hand side)

equals - ACy - BDx < 0 for all x, y > 0. This implies that any attractor of the dynamics has measure zero. Thus, a strange attractor exists when the flow is wrapped around a bounded region indefinitely by using unlimited reinforcements.

A First Integral (Constant of Motion)

A first integral (constant of motion) for the system of linear differential equations is $I_0(x,y) = ay^2 - bx^2$. The curve defined by $I_0(x,y)=C$ (constant) is an orbit of a solution of the differential equation. The orbit through (0,0) is a straight line $y = \sqrt{(b/a)}\, x$, $x \geq 0$. By integrating equations (3) an analytic first integral $I(x,y)$ for the nonlinear differential equations is

$$I(x,y) = (B/C)x - (A/D)y - (B/C^2)ln(1+Cx) + (A/D^2)ln(1+Dy).$$

If $Cx = Dy$ and $(B/C)x = (A/D)y$, then the orbit that passes through the origin is a straight line. In any event, setting $I(x,y) = 0$ and finding the orbit for $x, y > 0$ yields a curve that separates the quadrant into two regions where a win occurs without reinforcement. In order to specify the reinforcement criterion above, that $r > s > \sqrt{(B/A)}$, we examine the direction of approach to the origin (0,0) of the integral curve defined by $I(x,y) = 0$ for $x, y > 0$. We want to find the tangent direction of the curve at (0,0) that is the limiting direction of the vector field (3) as the curve approaches (0,0). The tangent direction is

$$dy/dx = Bx(1+Dy)/[Ay(1+Cx)]. \tag{4}$$

A simple procedure yields the limiting direction along the curve defined by $I(x,y) = 0$. We set $y = \alpha x$ and substitute into (4). This yields at $x = 0$ the expression $B/A = \alpha^2$. With limited reinforcements, nonmonotonicity of battle outcomes is possible in this model. With unlimited reinforcements, chaos occurs.

A Discrete Model Derived from the Nonlinear Model

By using the same technique to write down a replacement of the derivative as a finite difference, a discrete dynamical system is obtained for the nonlinear differential equations (3)

$$x_{n+1} = x_n - hAy_n(1+Cx_n)$$
$$y_{n+1} = y_n - hBx_n(1+Dy_n) \tag{5}$$

This system of difference equations is nonlinear in x_n and y_n. By adding criteria for reinforcements the system of difference equations and auxiliary conditions is nonlinear

in two distinct ways: the difference equations are nonlinear and the reinforcement strategy causes wraparound—a nonlinear effect.

Chaos and Nonmonotonicity

We define a mapping F in two dimensions by

$$F: (x, y) \rightarrow (x - hAy[1+Cx], y - hBx[1+Dy]) \qquad [6]$$

By differentiating (6) we obtain the Jacobian matrix DF(x,y). Finally, by computing the determinant $|DF(x, y)|$ we obtain the Jacobian J(F) of the mapping F

$$J(F) = 1 - h^2AB - (hBD + h^2ABC)x - (hAC + h^2ABD)y. \qquad [7]$$

For x, y > 0, J(F) < 1. This shows that the nonlinear "area fire" terms in the difference equations (5) only increase the rate of area decrease for h > 0. Thus, for all cases in which h, x, y > 0, the measure of an attractor equals 0.

Use of the equations in (7) shows that a positive Lyapunov exponent exists for bounded orbits of the nonlinear difference equations (5). Let $\Lambda_{\pm}(x,y)$ denote eigenvalues of the derivative matrix of (6). The multipliers are computed from

$$\Lambda^2 - [2 - h(ACy + BDx)]\Lambda + [1 - h(ACy + BDx) - h^2AB(1+Cx+Dy)] = 0.$$

For A, B, C, D, h, x, y > 0, Λ is expressed by

$$\Lambda = 1 + \tfrac{1}{2}h(ACy + BDx)[-1 \pm \sqrt{(1 + [4AB(1+Cx+Dy)/(ACy+BDx)^2])}].$$

This demonstrates that $\Lambda_+ > 1$ and $\Lambda_- < 1$ for all choices of A, B, C, D, h, x, y > 0. For C=D=0 Lyapunov multipliers of the linear system are recovered from the quadratic equation $\Lambda_{\pm} = 1 \pm h\sqrt{(AB)}$. This demonstrates a positive Lyapunov exponent for every orbit of the system (5) that is bounded away from (0,0). This in turn implies that chaos exists in the discrete dynamical system (5).

Figure 2 is a three-dimensional Battle Outcome Diagram for initial strengths of Red (fixed per panel) and Blue (increasing along horizontal). It shows nonmonotonicity as the effect of nonlinearity as the strengths of area fire terms are increased from zero to positive in 0.01 increments of the coefficients. A nonlinearity effect is expected. As the initial strength of Blue force is increased, area fire is expected to be significant. For fixed initial strength of Red, follow Blue's initial strength moving from left to right along a horizontal line. At first Red wins. As Blue's initial strength increases, Blue

Figure 2. Battle outcome diagram—initial strength of blue vs nonlinearity (red initial strength is fixed per panel).
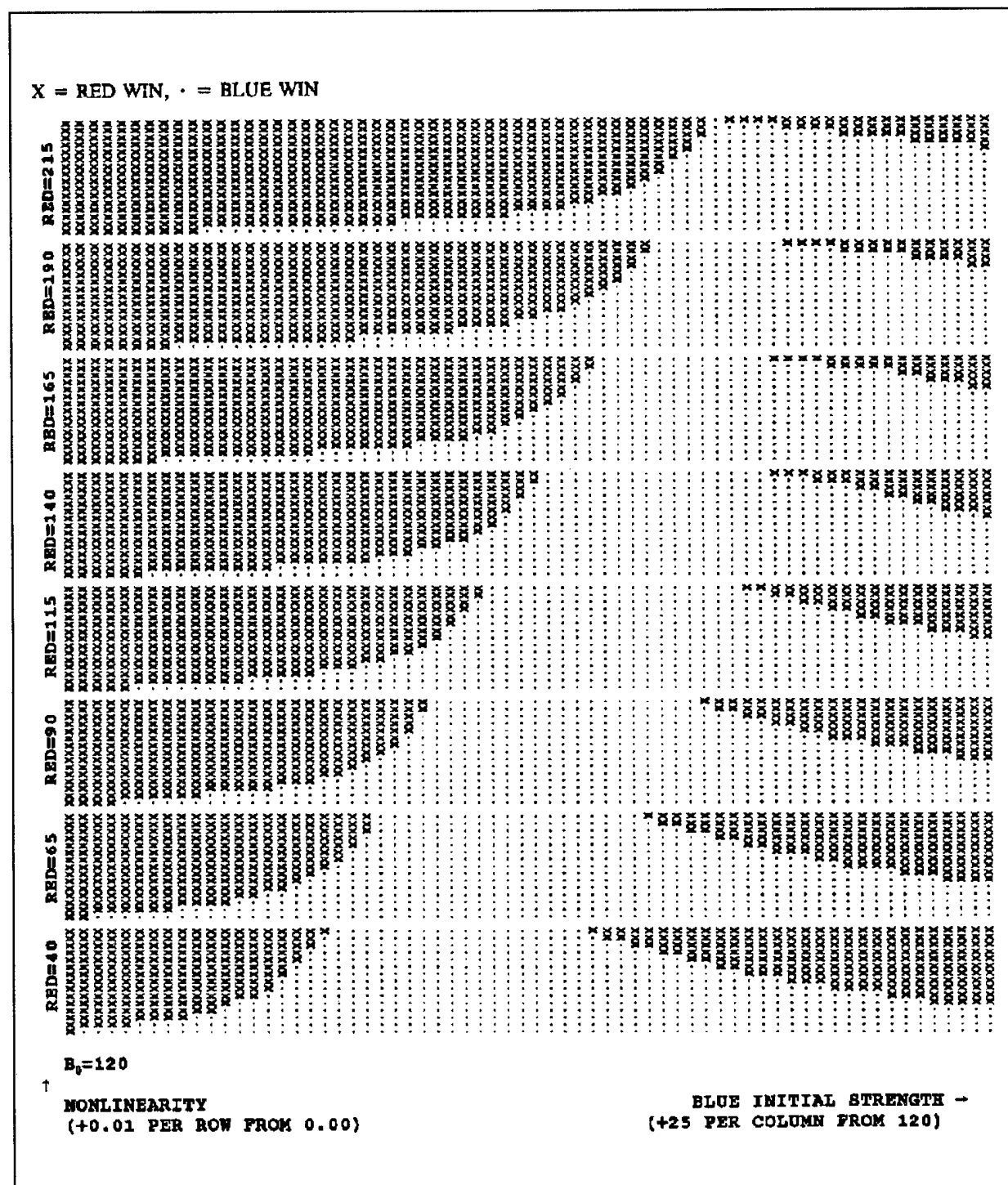
wins. Eventually, area fire gives Red a significant advantage. As Blue's strength increases further, Red wins.

Conclusion

The dynamics of simple models of attrition and reinforcement show chaotic behavior when computed as *difference equations*. With the addition of nonlinear area fire terms, the nonlinear difference equations show nonmonotonicity of battle outcomes.

7 Summary

Computer Arithmetic

It is important to understand the precisions and roundings protocols of the computer system being used in combat modeling and simulation. IEEE standards give 12 combinations of precisions and roundings in common use. These are single, double, and extended double precisions and round up, round down, round to zero, and round to nearest roundings. These 12 combinations of precisions and roundings give a "poor man's test" of sensitivity.

Computer Arithmetic Effects

Computer arithmetic can affect the outcome of computer simulations that are sensitive to input data and branchings on thresholds. Data entered in decimal form are converted to binary form and stored in memory. Thus, tiny changes can occur even as data are entered. Depending upon the design of thresholds for decisionmaking, these tiny changes can affect branchings on thresholds and the outcome of combat simulations. For example, entering a unit's location as a single precision value into double precision can cause the unit's location to go unrecognized. For example, $x = 0$ (single precision) when entered into double precision can be stored as 0 (single precision) + ? in the remaining double precision bits. When compared to a double precision $X = 0$ by an if-then statement such as "if $x = X$, then do...," it can happen that the comparison finds $x \neq X$. Such computer arithmetic effects must be guarded against in the design of combat simulations. The strategies for eliminating or reducing structural variance—unusual or unexpected results—are found in Chapter 2, Table 1. Model design and sensitivity considerations are given in Tables 2 and 3.

Chaos and Nonmonotonicity

Discrete algorithms can cause chaotic behavior even when the differential equation models from which the discrete algorithms are designed have none. One way in which the chaotic effects are introduced is with branching on thesholds. This is observed easily in the models of attrition and reinforcement in Chapter 6. One must guard

against the introduction of instabilities and chaos into the design of combat models. Causes of instabilities are summarized in Chapter 2, Table 4. A design solution to this problem is to ensure that there is a global time used throughout a simulation, especially when discrete event simulations are present in the combat model.

References

ANSI/IEEE Standard 754 on Binary Floating Point Arithmetic, 1985, Global Engineering Documents, Irvine, CA.

Davis, P.K., 1992a, "Dynamic Instability," in Technical Forum on Nonlinear Behavior in Combat Simulations, PHALANX, December, and Private Communication on Structural Variance.

Davis, P.K., 1992b, "Generalizing Concepts and Methods of Verification, Validation, and Accreditation (VV&A) for Military Simulations," R-4249-DR&E, RAND Corporation, Santa Monica, CA.

Davis, P.K., 1992c, "An Introduction to Variable Resolution Modeling," WD-6086-DARPA, RAND Corporation, Santa Monica, CA.

Davis, P.K., and D. Blumenthal, 1991, "The Base of Sand: A White Paper on the State of Military Combat Modeling," N-3148, RAND Corporation, Santa Monica, CA.

Dewar, J., J. Gillogly, and M. Juncosa, 1991, "Non-Monotonicity, Chaos and Combat Models," R-3995-RC, RAND Publication, Santa Monica, CA.

Dewar, J., J. Gillogly, and M. Juncosa, 1990, "Non-Monotonicity, Chaos and Combat Models," preprint submitted to Proceedings of the 59TH MORS Symposium, The Military Operations Research Society, Alexandria, VA.

Fishman, G., 1978, *Principles of Discrete Event Simulations*, John Wiley and Sons, New York.

Fishman, G., 1973, *Concepts and Methods in Discrete Event Digital Simulation*, John Wiley and Sons, New York.

Fishman, G., and L. Moore, 1986, "An Exhaustive Analysis of Multiplicative Congruential Random Number Generators with Modulus $2^{31} - 1$," *SIAM J. Sci. Stat. Comput.*, 7, pp 24-45.

Hillestad, R.J., J. Owen, and D. Blumenthal, 1992, "Experiments in Variable Resolution Modeling," WD-6066-DARPA, RAND Corporation, Santa Monica, CA.

Hillestad, R.J., and M. Juncosa, 1992, "Cutting Some Trees to See the Forest: On Aggregation and Disaggregation in Combat Models," WD-6065-DARPA, RAND Corporation, Santa Monica, CA.

Hodges, J., and J. Dewar, 1992, "Is It You or Your Model Talking? A Framework for Model Validation," R-4114-RC/AF, RAND Corporation, Santa Monica, CA.

Khinchin, A. Ya., and B.V. Gnedenko, 1962, "An Elementary Introduction to the Theory of Probability," 5TH edition, translated by L.F. Boron, Dover Publications, New York, p 121.

Kuhn, T., 1970, "The Structure of Scientific Revolutions," Princeton Univ. Press, 2ND edition. See "Postscript."

Mann, S.R., 1992, "Chaos Theory and Strategic Thought," *Parameters*, U.S. Army War College Quarterly, Autumn, pp 54-68.

McCauley, J.L., Jr., and J.I. Palmore, 1986, "Computable Chaotic Orbits," Physics Letters A, 115, pp 433-436.

Middleton, G.V., ed., 1991, "Nonlinear Dynamics, Chaos and Fractals with Applications to Geological Systems," Short Course Notes, v. 9, Geological Association of Canada.

Palmore, J.I., 1996, "Dynamical Instability in Combat Models: Ccomputer Arithmetic and Mathematical Models of Attrition and Reinforcement," *Military Operations Research*, vol 2, no. 1 (Spring 1996), pp 45–52.

Palmore, J.I., 1994, "More on the Subjectivity of Computers," *Communications of the ACM*, vol 37, no. 6, pp 89-90.

Palmore, J.I., 1993a, "Verifying Complex Discrete Event Simulations" *in* Proceedings of the 32nd Annual U.S. Army Operations Research Symposium (AORS XXXII), Fort Lee, VA, 12–14 October 1993, vol 2, pp G3-G16.

Palmore, J.I., 1993b, "Verification, Validation, and Visualization of Dynamical Processes in a Parallel Computing Environment," *in* High Performance Computing Symposium 1993, Grand Challenges in Computing Simulation (HPC '93), Society for Computer Simulation, Arlington, VA, 29 March–1 April 1993, pp 197-202.

Palmore, J.I, 1993c, "Verifying Discrete Event Simulations Containing Dynamical Systems," *in* HPC '93, Society for Computer Simulation, Arlington, VA, 29 March–1 April 1993, pp 231-236.

Palmore, J.I., 1992a, "Dynamical Instability in Combat Models," *in* Technical Forum on Nonlinear Behavior in Combat Simulations, PHALANX, December.

Palmore, J.I., 1992b, "Verification and Validation Methodology for Combat Models," *in* Proceedings of AORS XXXI, U.S. Army Operations Research Symposium, Fort Lee, VA, 1992.

Palmore, J.I., 1992c, "Variable Resolution Modeling in Mathematics," *in* Proceedings of the Conference on Variable Resolution Modeling, RAND Corporation, Santa Monica, CA.

Palmore, J.I., 1992d, "Analysis and Verification and Validation of Complex Models," *in* Proceedings of the Summer Computer Simulation Conference, The Society for Computer Simulation, Reno, NV, pp 139-143.

Palmore, J.I., 1992e, "Dynamical Instability in Combat Models" *in* Proceedings of the 60th Military Operations Research Symposium, U.S. Naval Postgraduate School, Monterey, CA, 22–25 June 1992, pp 23-42.

Palmore, J.I., 1991a, "A Review of Nonlinear Dynamics, Chaos, and Fractals," J. Geological Education, vol 39, pp 393-397.

Palmore, J.I., 1991b, "Instability of Computer Simulations of Combat Models," *in* Proceedings of AORS XXX, U.S. Army Operations Research Symposium, U.S. Army Training and Doctrine Command Analysis Command, Fort Lee, VA, III, pp 170-182.

Palmore, J.I., 1991c, "Verification and Validation of Computer Simulations of Deterministic Dynamical systems," *in* Proceedings of the Summer Computer Simulation Conference, The Society for Computer Simulation, Baltimore, MD, pp 26-31.

Palmore, J.I., 1990, "Computer Arithmetic and Dynamical Simulations," *in* Proceedings of AORS XXIX, U.S. Army Operations Research Symposium, U.S. Army Materiel Command, Alexandria, VA, II, pp 363-372.

Palmore, J.I., 1988, "Exploring Chaos: Science Literacy in Mathematics," *in* Proceedings of the Symposium on Science Learning in the Informal Setting, The Chicago Academy of Sciences, Chicago, IL, pp 183-200.

Palmore, J.I., S.A. Burns, and H.E. Benzinger, Jr., 1988, "Ecology Models and Newton Vector Fields," *J. Mathematical Biosciences,* vol 90, pp 221-232.

Palmore, J.I., and C. Herring, 1995, "Random Number Generators are Chaotic," *Communications of the ACM,* vol 38, no. 1, pp 121-124.

Palmore, J.I., and C. Herring, 1990, "Computer Arithmetic, Chaos and Fractals," Physica D, pp 99-110.

Palmore, J.I., and J.L. McCauley, Jr., 1989, "Temperature and Initial Conditions on Strange Repellers," *Int. J. Modern Phys. B,* vol 3, pp 1447-1453.

Palmore, J.I., and J.L. McCauley, Jr., 1987, "Shadowing by Computable Chaotic Orbits," Physics Letters A, vol 122, pp 399-402.

Singleton, G., 1992, "Chaotic Behavior in Battle Outcomes," Draft Report for U.S. Army Training and Doctrine Command, Fort Monroe, VA.

Tsai, H.K., and J. Ellenbogen, 1992, "Bounding Potentially Pathological Nonlinear Behavior in Combat Models and Simulations," *in* Technical Forum on Nonlinear Behavior in Combat Simulations, PHALANX, December.

U.S. Army, 1992, Army Regulation 5-11, Army Model and Simulation Management Program, Office of the Deputy Under Secretary of the Army for Operations Research.

U.S. General Accounting Office, 1992, "Patriot Missile Defense, Software Problem Led to System Failure at Dhahran, Saudi Arabia," GAO/IMTEC-92-26, Washington, D.C.

U.S. General Accounting Office, 1988, "DoD Simulations: Improved Assessment Procedures Would Increase the Credibility of Results," GAO/PEMD-88-3, Washington, D.C.

U.S. General Accounting Office, 1980, "Models, Data and War: A Critique of the Foundation for Defense Analysis," GAO/PAD-80-21, Washington, D.C.

Westenhoff, C.M., ed., "Oils for the Friction of War," *in* Military Air Power: The Cadre Digest of Air Power Opinions and Thoughts, Air University Press, Maxwell Air Force Base, AL, 77-79.

Glossary

AGGREGATE — To assemble; to combine into a mass. An aggregate model uses fewer entities to represent a real-world activity than a *disaggregated model*. This term is used in *variable resolution modeling*.

CHAOS — Chaos is defined historically as great disorder and confusion. Technical definitions of chaos attempt to make precise the dynamical mechanisms that cause disorder. The paradigm for chaos is the Bernoulli shift. For a Bernoulli shift the dynamical mechanisms of stretching and wraparound apply. Any dynamics that entails stretching, folding, and wraparound is a candidate for chaos. For differentiable systems a criterion for the existence of chaotic dynamics is a positive Lyapunov exponent. Technical definitions vary so the reader and writer must take care [McCauley and Palmore 1986; Palmore and McCauley 1989, 1987; Palmore et al. 1988; Palmore 1988].

COMBAT MODEL — A combat model is a nonlinear deterministic model of decision-making processes that deals with attrition of opposing forces.

COMPLEX MODEL — A complex model is one that is designed using several different interacting modeling paradigms. A complex simulation is an implementation of a complex model's dynamics. Interactions between the paradigms of a complex simulation occur at interfaces. A combat simulation is the implementation of a complex model that has an ambient discrete event simulation in which dynamical systems are embedded and which relies heavily on computer arithmetic.

COMPUTATIONAL CHAOS — When computations require iteration of algorithms there can be chaos in the process. The algorithms are discrete dynamical systems with basins of attractions of solutions. Two or more solutions can be attractors of an algorithmic iterative process. Chaos exists on the boundary of basins of attraction.

COMPUTER ARITHMETIC — Real number arithmetic uses four binary operations: addition, subtraction, multiplication, and division by nonzero numbers. As iterative processes, multiplication is iterative addition and division is iterative subtraction. Computer arithmetic is the implementation of arithmetic as finite precision arithmetic with restricted exponents.

CONJUGACY — Invertibility of aggregation and disaggregation. Conjugacy allows reversing the action of aggregation and disaggregation.

CONSISTENCY — This concept from variable resolution modeling pertains to aggregation and disaggregation. Consistency in the aggregate has to do with maintaining simulation runs of aggregated models to within a given tolerance.

DISAGGREGATE — To disassemble; to separate into particulars. This term is used in variable resolution modeling.

DISCRETE EVENT SIMULATION — "By a discrete event simulation we mean one in which a phenomenon of interest changes value or state at discrete moments of time rather than continuously with time" [Fishman 1973, 1978].

DYNAMICAL PROCESS — Historically, a process is a "phenomenon that proceeds according to time" [Khinchin 1962]. Dynamics can be studied either as deterministic dynamics or stochastic processes. Stochastic processes contain random variables and rules that describe how they evolve in time. Deterministic dynamics can be identified as dynamical systems that contain no stochastic variables.

DYNAMICAL SYSTEM — Dynamical systems model dynamical processes deterministically. Dynamical systems have global time variables and deterministic laws of evolution for the state variables. A discrete dynamical system is defined by a function f mapping a space X into itself. An example is the logistic map $f(x)=4x(1-x)$, x in $0 \leq x \leq 1$. Each iteration is a single time step for this dynamical system.

DYNAMICAL INSTABILITY AND CHAOS — Dynamical instability and mathematical chaos are defined with respect to dynamical systems.

EVENT — Events mark the beginning or end of activities. An event has no time frame associated with it. Thus, discrete event simulations have no global time variables.

FINITE PRECISION (COMPUTER ARITHMETIC) — Finite precision refers to the number of significant bits (in base 2) that represent numbers in a given system of computer arithmetic. IEEE/ANSI Standard 754 specifies single precision as 24 bits (in a 32 bit word), double precision as 53 bits (in a 64 bit word) and extended precision as 64 bits (in an 80 bit word).

FUNCTION (MAPPING) — A function is a correspondence between points in the domain space of the function and the range space of the function. Numerical functions are real

valued functions of a real variable: $f(x) = x^2$, for all x, is a numerical function. The domain and range spaces are the real numbers.

GLOBAL — Global defines the largest domain of an entity. Global entities are accessible from all parts of a computer program. Contrast "global" with "local." A local entity can be accessed only in the module in which it is defined.

GLOBAL DIVERGENCES — Global divergences refers to divergences of simulation runs from a base case. This term is used here to refer to simulation behavior observed in sensitivity analyses.

GLOBAL TIME VARIABLE — Global variables are accessible from all parts of a program. A global time variable is a simulation's reference time. It is found in dynamical systems simulations in laws of evolution. Local time variables, used in computing, do not necessarily match a global time. An example of a mismatch is found in GAO 1992.

INTERFACES IN SIMULATIONS — A simulation interface is a location where information is translated from one form to another. One interface is the data entry location where decimal representations are converted to binary for use by the machine. Another is the output location where binary information is converted to decimal format for use by modelers. Module interfaces exist and are well defined in object oriented paradigms.

MODEL — A model consists of objects, attributes of the objects, and relationships between objects. If the model is dynamical, then laws of evolution for the objects are given. The model is an abstraction of real world entities, their attributes, and relationships between entities. A correspondence between real-world entities and model objects is established by functional requirements of the model concept.

MODEL OF COMPUTATION — A model of computation is a complete specification of the discrete setting in which a computation proceeds. For example, a model of computation for a heat transfer problem is a statement of the mathematical problem, the algorithms that are used to compute solutions, and the model of arithmetic to be used in computing.

MODEL OF COMPUTER ARITHMETIC — A model of computer arithmetic is finite precision arithmetic in a machine environment that results by choosing among available precision and rounding options, data types, and other specifications.

NONLINEAR EFFECTS — Nonlinear means "not linear." The linear numerical functions have the form ax where the multiplier a is a real number and x is any real number.

In particular, Bernoulli shift x → ax mod 1, 0≤X<1, is nonlinear because the modulo operation forces the interval to be wrapped around itself. The shift is not linear.

NONMONOTONICITY — For systems in which an order can be placed on outcomes, a question is whether outcomes change monotonically with input. Where nonmonotonicity exists in a system, an increase in input does not yield a monotone change in outcome. A win/loss picture such that slight increases of input can change the outcome from win to loss is an example of nonmonotonic behavior. In this type of case input may be a measure of the resources available.

OBJECTS — Objects and entities are the subjects of modeling. A model of real-world activities is a correspondence between the model's objects and real world's entities. This need not be a 1-to-1 correspondence. Model objects may represent aggregated real-world entities as in battalions, convoys, companies.

ORBIT OF A DYNAMICAL SYSTEM — When a dynamical system is defined by ordinary differential equations with initial value problems, an orbit is the path taken by a solution curve. It is the set of points in the domain that is the image of the solution as a function of time. In general, the orbit of a dynamical system is the path followed by a solution.

PARADIGM — Thomas Kuhn defines the modern meanings of "paradigm" in his book *The Structure of Scientific Revolutions* [Kuhn 1970]. A paradigm is a world view of a selected community. A world view is given by the ideas, methods, and values of the selected community. The paradigm of discrete event simulations is defined by the ideas, methods, and values of the simulation community; the paradigm of dynamical systems is defined by the mathematics, physical sciences, and engineering community; the paradigm of computer arithmetic is defined by the electrical engineering, computer engineering, and computer science community.

RESOLUTION — A set of scales: the smallest scale(s) at which information is recognized, or the smallest scale(s) that define details in a model or simulation.

SCALABILITY — Dynamic scalability is "the ability to accommodate large, dynamic changes in size and complexity within models and simulation systems." Changes contemplated are in cardinality (the number of objects), granularity (the fidelity and resolution of objects and environment), heterogeneity (diversity of objects and environment), variability (allowing cardinality and granularity to differ in time and location) [DMSO Focused Call For Proposals, 1992].

SIMULATION (COMPUTER) — A simulation implements a model. Computer simulations implement the models on digital computers.

STRUCTURAL STABILITY — Structural stability refers to the robustness of a model or simulation as the form of the model is changed. Small changes do not affect the results of structurally stable computer simulations. Even when a computer simulation produces chaotic results, if the simulation is structurally stable, then chaos persists as small changes are made to the simulation.

STRUCTURAL VARIANCE — Global divergences of simulation runs. Compare "structural variance as manifested, e.g., by peculiar sensitivity results and major changes in results if one shifts from one computer to another" [Davis 1992b].

SENSITIVITY ANALYSIS — A sensitivity analysis is performed by running a simulation with a variety of sets of initial data to detect structural variance in output. It is used (1) to test for desired sensitivity of a simulation as a response to input data changes and (2) to test for unwanted sensitivity. Data sets can be generated by sensitivity analysis by iterating the initial data until a desired effect is achieved.

VARIABLE RESOLUTION — Two or more sets of scales. Variable resolution arises from the smallest scales in each set of scales at which information is recognized [Palmore 1992c].

VARIABLE RESOLUTION MODEL — Variable resolution models have several sets of scales. There are several dimensions to variable resolution modeling. Difficulties arise in variable resolution modeling when the sets of scales are dependent. This occurs in the system of astronomical constants in which theoretical dependencies between units exist [Davis 1992c; Hillestad et al. 1992; Hillestad and Juncosa 1992].

VV&A (VERIFICATION, VALIDATION, AND ACCREDITATION) — DoD terminology accepted by MORS [Davis 1992b; Hodges and Dewar 1992; Palmore 1992b, 1991c, 1990; U.S. Army 1992; U.S. GAO 1988, 1980]:

- VERIFICATION — The process of determining that a model implementation accurately represents the developer's conceptual description and specifications.
- VALIDATION — The process of determining the degree to which a model is an accurate representation of the real world in terms of the intended uses of the model.
- ACCREDITATION — An official determination that a model is acceptable for a specific purpose.

- MODEL CONCEPT — A statement of the content and internal relationships of what is to be represented in a model/simulation.
- MODEL DESIGN — A highly detailed description of a model/simulation, including descriptions of algorithms, logic and data flow, input and output data, and assumptions and limitations.
- MODEL CODE AND IMPLEMENTATION FOR HARDWARE/MAN IN THE LOOP SIMULATION — This is compilable computer code:

    COMPUTER MODEL/SIMULATION — This is the compiled and executed version of the code, including the specific computer hardware upon which that code is implemented. This is the final model/simulation to which VV&A applies. *Note that a distinction is made between the code and its implementation on specific hardware because that hardware (and associated representations of arithmetic) can affect results.*

    CODE VERIFICATION — Rigorous audit of the code (pseudo and compilable) to ensure proper implementation. This should be accomplished by both the model developer and an independent V&V (IV&V) agent.

    DATA VERIFICATION — Process of ensuring that source input data are (1) converted properly to model/simulation input data and (2) consistent with model concept and logical design.

    LOGICAL VERIFICATION — Process of ensuring, at each stage of implementation, that all assumptions and algorithms are consistent with the conceptual model.

DISTRIBUTION

Chief of Engineers
    ATTN: CEHEC-IM-LH (2)
    ATTN: CEHEC-IM-LP (2)
    ATTN: CECC-R
    ATTN: CERD-L

Defense Tech Info Center 22304
    ATTN: DTIC-FAB (2)

                8
             +30
         10/95